ПЕРЕДАВАЛЬНИЙ ПРИСТРІЙ СИСТЕМИ ТЕСТУВАННЯ КАНАЛУ ЦИФРОВОГО ЗВ'ЯЗКУ

Коцержинський Б. О., д.т.н., професор
Національний технічний університет України
«Київський політехнічний інститут», м. Київ, Україна

Для тестування каналу цифрового зв'язку розроблений на плісі передавальний пристрій забезпечує нескінчену циклічну передачу пакету бітових даних, який складається із 16-розрядної преамбули для визначення початку пакету у приймальному пристрої та псевдовипадкової послідовності довжиною 1023 бітів.

Генератор послідовності бітів реалізується за схемою (рис.1), де елементи X_{10} … X_1 створюють зсувний регістр [1], та працює за таким алгоритмом.

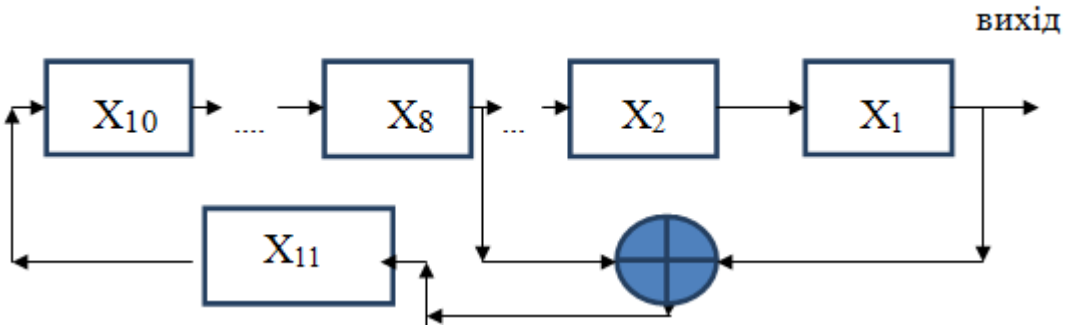

Рис. 1. Структурна схема генератора послідовності 1023 бітів

У регістр завантажується початкове значення і очищається лічильник кількості виведених бітів. Виводиться значення першого елемента регістра у послідовність. Виконується додавання за модулем 2 значень першого та восьмого елементів, результат якого зберігається у $X_{11.}$ Виконується зсув елементів регістра праворуч, елемент X_{10} заміщується значенням суми із X_{11}

І лічильник збільшується на одиницю. Відбувається перехід до виведення X_1, якщо не отримана потрібна кількість бітів.

Моделювання роботи генератора у середовищі C++ показало, що склад послідовності залежить від початкового значення регістра, у якому повинна бути хоча би одна 1 (можлива кількість різних 10-розрядних кодів становить 10!). Використання сзувного регістра потребує тільки першого початкового завантаження, після передачі 1023 бітів (одного періода) його початкове значення відновлюється, очищається 10-розрядний лічильник бітів. Створена C++ програма дозволяє перевірити незбіг вибраного довільно 16-розрядного коду преамбули із кодами послідовності.

Генерація псевдовипадкової послідовності може бути реалізована на послідовності тригерів або на сзувному регістрі, що виявилося краще.

Програмовний опис передавального пристрою мовою Verilog

```
module preaposl_top(clk,r,pout);
input clk,r;              // вхідні сигнали синхронізації  clk та запуску  r
output  reg pout;         // вихідна послідовність на pout
 reg T;
reg[15:0] codpre;         //16-розрядний регістр  для преамбули
reg[9:0] tmp;             // 10-розрядний регістр  генератора послідовності
parameter n=16;
integer i,k;              // лічильники
assign compare=(i==5'b10000);         //індикація найбільшого значення
                                      //лічильника бітів преамбули
assign compare1=(k==10'b1111111111); //індикація найбільшого значення
                                          //лічильника бітів послідовності

always @(posedge r or posedge clk)    //перемикання на зростаючих фронтах
                                      //сигналів запуску та синхронізації
if(r)                                 //починати при появі сигналу запуску
begin
codpre<=16'b1110101010101011;         //запис коду преамбули у сзувний ре-
гістр
i=1'b0;                   //початкове значення лічильника бітів преамбули
k=1'b1;                   //початкове значення лічильника бітів послідовності
tmp<=10'b0000000001; // початкове значення регістра генератора
end
else if(!compare)         //перевірка лічильника виведених бітів преамбули
begin                     // початок виведення бітів преамбули
pout=codpre[15];          //виведення старшого біта регістра преамбули
T=codpre[15];             //зберігання старшого біта регістра преамбули
codpre=codpre<<1;         //сзув бітів регістра преамбули ліворуч
codpre[0]=T;              //запис виведеного старшого біта  у молодший біт
i=i+1;                    //збільшення лічильника на 1
end                       //кінець виведення бітів преамбули
else if(!compare1)        // перевірка лічильника виведених бітів послідовності
begin                     // початок виведення бітів послідовності
 T=(tmp[0]^tmp[7]);       // функція зворотного зв'язку
  pout=tmp[0];            // виведення молодшого біта регістра генератора
tmp<=tmp>>1;              // сзув бітів регістра генератора праворуч
tmp[9]=T;                 // відновлення старшого біта регістра генератора
k=k+1;                    // збільшення лічильника на 1
end                       // кінець виведення послідовності
else begin                // перехід на виведення наступного пакету
i=1'b0;                   // відновлення лічильника
```

```
k=1'b1;                    // відновлення лічильника
tmp<=10'b00000000001;      //відновлення регістра генератора
endmodule                  //
```

Пристрій працює на плісах EP3C5E144C7, EP2C5AF25618, EP1K30T144-3.

Таким чином, для виведення пакету даних треба об’єднати преамбулу і послідовність у передавальному пристрої на плісі.

Симуляція можливих варіантів виведення коду преамбули у інтегрованому середовищі Quartus II фірми Altera для різних плісів показала, що найкращим способом є циклічне читання розрядів 16-розрядного зсувного регістра з кодом преамбули під контролем лічильника бітів, щоб після його розвантаження відновлювався його початковий код, а лічильник очищався. Передача починається з появою зростаючого фронту зовнішнього сигналу запуска і синхронізується вхідними сигналами тактової частоти.

Перелік посилань

1. Цифровые методы в космической связи / Под ред.С.Голомба. — М. : Связь, 1969. — 271 с.

Анотація

Представлені результати розробки передавального пристрою системи тестування каналу цифрового зв’язку на плісах. Моделювання на С++ та симуляція у середовищі Quartus сприяли вибору ефективних алгоритмів створення тестового пакету.

Ключові слова: генерація псевдовипадкової бітової послідовності. .

Аннотация

Представлены результаты разработки генераторного устройства системы проверки канала цифровой связи на плисах. Моделирование на С++ и симуляция в среде Quartus способствовали выбору эффективных алгоритмов создания тестового пакета.

Ключевые слова: генерация псевдослучайной битовой последовательности.

Abstract

The design results of FPGA generating device for digital communication channel test system are presented. C++ modelling and Quartus environment simulation stimulated the choice of test packet effective algorithms .

Keywords: pseudorandom bit sequence generation.